SINGLE MACHINE SCHEDULING WITH SEQUENCE DEPENDENT SETUP TIMES

A Thesis

by

Burak Lefkur

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Industrial Engineering

Özyeğin University
June 2021

SINGLE MACHINE SCHEDULING WITH SEQUENCE DEPENDENT SETUP TIMES

Approved by:

Assoc. Prof. Dr. Ali Ekici, Advisor
Department of Industrial Engineering
Özyeğin University

Assoc. Prof. Dr. Okan Örsan Özener
Department of Industrial Engineering
Özyeğin University

Prof. Dr. Serhan Duran
Department of Industrial Engineering
Middle East Technical University

Date Approved: 10 June 2021

To my family,

ABSTRACT

In this thesis, we study a single machine scheduling problem with sequence dependent-setup times where the jobs are processed in shifts with constant length. There is a certain amount of time between each pair of consecutive shifts for periodic maintenance, the machine does not work in maintenance but this time can also be used for the setup. Our goal is to complete the jobs in minimum amount of time. The setting considered in this thesis has applications in different industries including beverage and chemical industries.

The problem can be solved with a small number of jobs by using some mathematical models in computer environment. However, as the number of jobs increases, the difficulty of solving the problem increases exponentially. In order to overcome the problem, we proposed heuristic algorithms, and they provided us to find good solutions for the problem. The new heuristic algorithm is based on the First Fit Algorithm (FF) that is used for solution of bin packing problems.

Firstly, First Fit Decreasing (FFD) Algorithm has been applied to the problem, and jobs were assigned to shifts. For each shift in the result of FFD algorithm, used setup times were reduced by changing the orders of jobs in shift. The gaps occurred in some shifts with decrease in setup time used. All shifts were combined to fill in the gaps and FF algorithm was run. This process was repeated until there is no more improvement to be achieved for completion time of last job. This algorithm was named as Improved First Fit Heuristic (IFFH) algorithm. Then, according the result of IFFH algorithm, binary combinations of shifts and last shift were concatenated. The IFFH algorithm was repeated three times for the triple combinations of shifts. The minimum completion time was accepted as a solution. This algorithm was named

as Improved First Fit Heuristic-2 (IFFH-2) algorithm. Consequently, the algorithms were compared with benchmarks in the literature.

ÖZETÇE

Bu tezde, işlerin sabit uzunluğa sahip vardiyalarda yapıldığı, sıraya bağlı hazırlık sürelerine sahip tekli makine çizelgeleme problemini inceliyoruz. Her ardışık vardiya çifti arasında periyodik bakıma ayrılmış belirli bir süre vardır, makine bu süre içerisinde çalışmaz ancak bu süre hazırlık süresi olarak kullanılabilir. Amacımız tüm işleri minimum sürede tamamlamaktır. Bu tezde ele alınan ortam, içecek ve kimya endüstrileri dahil olmak üzere farklı endüstrilerde uygulamalara sahiptir.

Problem az sayıda iş için bilgisayar ortamında bazı matematiksel modeller kullanılarak çözülebilir. Ancak iş sayısı arttıkça problemi çözme zorluğu da katlanarak artmaktadır. Biz bu problemin üstesinden gelmek için bazı sezgisel algoritmalar önerdik, ve önerdiğimiz algoritmalar probleme iyi çözümler bulmamızı sağladı. Yeni sezgisel algoritmalar, kutulama problemlerinin çözümü için kullanılan İlk Sığan Algoritmasına dayanmaktadır.

Öncelikle probleme İlk Sığan Azalan Algoritması uygulandı ve işler vardiyalara atandı. İlk sığan azalan algoritmasının sonuçlarındaki her vardiya için, vardiyadaki işlerin sıraları değiştirilerek kullanılan hazırlık süreleri azaltıldı. Kullanılan hazırlık sürelerinin azalmasıyla birlikte vardiyada boşluklar oluştu. Boşlukları doldurmak için tüm vardiyalar birleştirildi ve ilk sığan algoritması çalıştırıldı. Bu süreç, son işin tamamlanma süresi için daha fazla iyileşme sağlanmayana kadar tekrar edildi. Bu algoritma, Geliştirilmiş İlk Sığan Sezgisel algoritması olarak adlandırıldı. Daha sonra, Geliştirilmiş İlk Sığan Sezgisel algoritmasının sonucuna göre, son vardiya hariç vardiyaların ikili kombinasyonları ile son vardiya birleştirildi. Geliştirilmiş İlk Sığan Sezgisel Algoritması, üçlü vardiya kombinasyonları için üç kez tekrarlandı. Minimum tamamlanma süresi çözüm olarak kabul edildi. Bu algoritma, Geliştirilmiş İlk Sığan

Sezgisel Algoritması-2 olarak adlandırıldı. Önerilen algoritmalar literatürdeki benzer örneklerle karşılaştırıldı.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my advisor, Dr. Ali Ekici, for his guidance and encouragement during my studies. I thank Vestel family for their patience and supports during my studies.

Finally, I am grateful to my parents, my sisters and my fiancée for their unconditional love and supports.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The beverage industry produces a variety of products which are in different flavors and formats, then they sent finished products to distinct points of sale. It cause the beverage industries to have quite high production levels. For the supply system to continue smoothly, production plants are forced to produce products effectively [13]. The beverage industry is the second largest sector in the European manufacturing industry in terms of value added. Therefore, efficient production planning in this sector enables producers to have more gain [11].

The bottling of beverages is a substantial process and its scheduling in the production line is difficult due to dependency between jobs. The reasons of this difficulty are that there are different types of label for some beverages, different molds of bottles in consecutive jobs, and preparation of syrup for different beverages. For example, if the glass bottle comes after the plastic bottle, the setup time is higher than if plastic bottle comes after the glass bottle. In order to overcome the problem, it is understood that a comprehensive scheduling is necessary.

There is a single machine for bottling of beverages and the production is carried out in shifts with constant length. The processing times of jobs are predetermined and do not change in production. There are sequence dependent setup times between jobs, this means that the setup times which will be used depends on the sequence of jobs. There is a certain amount of time between each pair of consecutive shifts which can also be used for the setup. Therefore, the setup time between last job of any shift and first job of next shift does not affect the completion time of all jobs. Our goal is to minimize completion times of all jobs by using comprehensive scheduling.

In order to make comprehensive scheduling, all variables that affects production are investigated.

There are uncertainties that make difficulties for manufacturers in production planning and meeting customer demands. One of the uncertainties of the scheduling is whether the demand type is deterministic. In this study, demand type is accepted as deterministic. There is another uncertainty about excessive demands which are ordered after the production cycle begins. Since unplanned orders can be received, the company try to satisfy customer demands with overtime. In this study, orders requested to be planned are determined before of production cycle.

The rest of the study is organized as follows: A literature review focused on investigated problem, is addressed in chapter 2. In chapter 3, definition of investigated problem and existing model are explained. In chapter 4, proposed solution approaches are presented. In chapter 5, computational results of proposed solution approaches are shown. Finally, in the last section the main conclusions are explained, and suggestions are made for future works.

CHAPTER II

LITERATURE REVIEW

In this section, literature review has been done. Single machine scheduling with setup times in production lines have been investigated. They are divided into two according to setup times of scheduled jobs: sequence independent and sequence dependent. Sequence independent setup times means that it depends on just processed job. However, sequence dependent setup times means that it depends on processed job and preceding job. [1] This thesis deals with a variant of single machine scheduling problems with sequence dependent setup times.

Glassey [9] scheduled production on a single machine for n product in a finite horizon by using dynamic programming formulation of the shortest route problem. The objective of scheduling problem is meeting deliveries on time and minimizing the number of change-overs of the machine one product to another.

Shwimer [15] used branch and bound algorithm to schedule jobs that are sequence independent. Objective function of the Shwimer's problem was minimize the total penalty cost caused by tardiness of jobs.

Gupta and Kyparisis [12] classified single machine scheduling problems as deterministic and stochastic in terms of objective functions and constraints. In deterministic problems, there are no external events that affect the sequencing. Objective functions include: Completion time with due dates, completion time without due dates, set up time, variable capacity. In stochastic problems, external events and breakdowns are

considered for same objectives.

Biskup and Cheng [5] studied about the scheduling problem on a single machine, where processing times are compressible as a linear cost function. They demonstrated that optimal sequencing and optimal due dates can be found by considering the problem as an assignment problem for n independent and simultaneously available jobs on a single machine.

Baker and Magazine [3] examined optimal solution procedure about single machine scheduling to minimize maximum lateness of jobs. They developed a solution approach for batching and sequencing model, where it includes job due dates and setup times. The solution algorithm works with branch and bound procedures. They studied about composite jobs, it provides them to decrease the problem size. They found that computational difficulty in that the best optimization algorithms solve the problem at most 60 jobs.

Sourd and Kedad [16] addressed the single machine scheduling problem for distinct due dates, earliness costs and tardiness costs. They suggested the lower bound that based on decomposition of each jobs to single operations and used branch and bound algorithm to solve problem

Ronconi and Kawamura [14] developed branch and bound algorithm to solve the single machine scheduling problem with a common due date aiming to minimize earliness and tardiness penalties

Gavet [8] studied about single production facility and he discussed the problem as traveling salesman problem. The objective of problem is to minimize the facility time

and setup time among jobs. He examined the performance of three heuristics rules for scheduling n jobs, which are 'next best', 'next best prime' and 'next best double prime'. Scheduling in next best algorithm begins with initial job and goes on job which have minimum sequence dependent setup time. Scheduling in next best prime begins with initial job, and for other jobs next best algorithm are repeated for n-1 times. Scheduling in next best double prime begins with initial job, then next best algorithm are used after subtracting the minimum value of the possible setup times before any job from these possible setup times. The conclusion of Gavet's study is heuristic methods give a better solution than random sequencing, moreover next best prime gives a better solution than next best algorithm. Next best double prime algorithm most probably gives a better solution than next best prime algorithm, but in the some examples it is less efficient, due to chance effect on setup times.

Sun, Noble and Klein [17] studied single machine scheduling problems with sequence dependent setup times, where the objective is minimizing the weighted sum of squared tardiness. They formulated sequence dependent setup times as capacity constraints. They relaxed capacity constraints by using Lagrangian Relaxation techniques. They created primal and dual problems, then solved them by using sub-gradient technique. After that, they used three way swap for jobs according to their starting times to enable improvement for objective. The conclusion demonstrated that the approach used gives a better solution compared with Earliest Due Date, Apparent Tardiness Cost with Setups rule, Tabu Search and Simulated Annealing.

Gagne, Price and Gravel [7] stated that single machine scheduling is NP-hard problem. They proposed Ant Colony Optimization heuristics to overcome the huge size of problems faced in industries. They modified existing Ant Colony Optimization

method by using look ahead information. They used look ahead information in selection of next job in solution with the help of statistical tests, where the objective is minimizing total tardiness. A comparison with a genetic algorithm, a simulated annealing approach, a local search method and a branch-and-bound algorithm demonstrated that Ant Colony Optimization which they used is competitive for huge size of problems.

Bigras, Gamache and Savard [4] studied about scheduling problems on a single machine in a sequence dependent setup environment. They proposed several stronger integer programming formulations. Although, the calculation of their LP relaxation is more time consuming, experiments demonstrated that they are beneficial. By using LP relaxation, authors proposed exact Branch and Bound algorithm to solve problem. The algorithm can solve problems which have 45-50 jobs.

Arroyo, Nunes and Kamke [2] investigated single machine scheduling with sequence dependent setup times for the objective of minimizing total tardiness. They proposed Iterative Local Search (ILS) Heuristic that generate initial solution by using Greedy Randomized Adaptive Search Procedure. They compared ILS algorithm with GRASP algorithm and Ant Colony Optimization algorithm. The study showed that these algorithms were better than algorithms in the literature. Moreover, ILS is more efficient and competitive for up to 85 jobs.

Coffmann, Garey and Johnson [6] stated that worst case ratios of Next Fit (NF), Best Fit (BF), First Fit (FF), Next Fit Decreasing (NFD), Best Fit Decreasing (BFD) and First Fit Decreasing (FFD) algorithms are respectively as follows, 2.0, 1.7, 1.7, 1.69, 1.22 and 1.22. The worst case ratio for these algorithms equals number of required bin for related algorithm divided by optimal required number of bins. The study

demonstrated that FFD algorithm gives a better solution in comparison with other algorithms.

Gonzalez and Framinan [10] studied about single machine scheduling with periodic maintenance and their objective was minimizing make-span. They thought that the problem is similar to the classical bin packing problem. Then, they propose new procedures to solve problem, and then they compared them between each other. They are compared 15 existing heuristic algorithm with 3 proposed algorithms. Their proposed algorithms are based on simple local search. FF, BF and NF algorithms are firstly applied on sequence which is created by Longest Processing Times First (LPT) method. After that, a random job is picked and it is put in a different place by starting from first sequence. Then, FF, BF and NF is applied on new sequence. If there is an improvement on make-span, algorithm continue with picking another job and putting it to new place, otherwise, picked job is placed to next sequence.

They studied with different number of jobs starting from 10 jobs to 300 jobs for the problem. The processing times of jobs and size of blocks uniformly distributed. They classified instances to two set according to tightness size of blocks. Relative percentage deviation from optimal solution is higher when size of blocks is tighter. Moreover, they demonstrated that their new algorithms gives a better solution than other existing heuristic algorithms.

We are inspired by this article, developed new heuristic algorithms and compared them with classical bin packing algorithm FFD. We chose FFD algorithm to improve because its relative percentage deviation is smallest according to other heuristics in this article. There are not sequence dependent setup times in this article, but the investigated problem includes. Moreover, there are shifts instead of blocks, and length of them is fixed in the investigated problem. We were studied with high number of jobs starting from 50 to 500. Our study does not include optimal solutions, there

are lower bounds instead of it to calculate relative percentage deviations. We found better solutions than FFD as in the article, moreover improvement rates are higher than article in our study, because we evaluated opportunities to reduce used setup times.

As a result of discussion of all articles in literature review, we see that there are many different studies for single machine scheduling and there are many solution approaches. Some of articles demonstrated comparison of existing models, some of them proposed new heuristic algorithm. The difference of our study from them is that we dealt with problem which has sequence dependent setup times as bin packing problem, we proposed heuristic approach based on bin packing algorithms.

CHAPTER III

PROBLEM DEFINITION

In this chapter, investigated problem and existing model are explained. Firstly, the setting considered in this study is specified with parameters and the objective of the problem is explained clearly. Then, existing model of the problem is demonstrated. The sets, decision variables, parameters and mathematical model of existing model are shown in different sections. Finally, all constraints of the mathematical model are explained in detail.

There are n jobs which have sequence dependent set-up times between each other. The objective of the problem is to complete all jobs in minimum amount of time. In other words, we want to minimize the make-span. Shifts each with a constant length (in terms of time) are scheduled to process jobs. Total length of a shift is denoted by T. The jobs have certain processing times. The processing times are denoted by P_i. There is a sequence dependent set-up time between each of pair jobs that are processed consecutively. The setup times between jobs i and j are denoted by H_{ij}. Besides, there are breaks between each pair of consecutive shifts for periodic maintenance, which can be used for setup times. It means that the setup time between the last job of any shift and the first job of next shift can be ignored due to completing them on breaks. Therefore, there is not any setup times for first jobs of each shift.

3.1 Existing Model

3.1.1 Sets

N: Set of jobs, N={1, ... ,n}

K: Set of shifts, K={1, ... ,n}

L: Set of sequences, L={1, ... ,n}

3.1.2 Decision Variables

$$x_{ikl} = \begin{cases} 1, & \text{if job i assigned to sequence l of shift k} \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in N \ \ \forall k \in K \ \ \forall l \in L$$

$$y_{ijk} = \begin{cases} 1, & \text{if job i processed before job j on shift k} \\ 0, & \text{otherwise} \end{cases} \qquad \forall i, j \in N \ \ \forall k \in K$$

$$z_k = \begin{cases} 1, & \text{if there is a production on shift k} \\ 0, & \text{otherwise} \end{cases} \qquad \forall k \in K$$

S_{ikl} = Starting time of job i which is in kth shift and lth sequence.

C_{ikl} = Completion time of job i which is in kth shift and lth sequence.

C_{max} = Makespan

3.1.3 Parameters

H_{ij}= Sequence dependent setup time between job i and job j.

P_i= Processing time of job i.

T= Length of Shift

M= Big Number

3.1.4 Model

Objective Function:

$$min \ C_{max}$$

subject to:

$$\sum_{K} \sum_{L} x_{ikl} = 1 \ \ \forall i \in N \tag{1}$$

$$\sum_{I} x_{ikl} \leq 1 \ \ \forall k \in K, \forall l \in L \tag{2}$$

$$x_{ik,l-1} + x_{jkl} - y_{ijk} \leq 1 \ \ \forall k \in K, \forall l \in \{L/1\} \tag{3}$$

$$C_{ikl} \geq S_{ikl} + P_i * x_{ikl} \ \ \forall i \in N, \forall k \in K, \forall l \in L \tag{4}$$

$$S_{jkl} - C_{ik,l-1} + M * (1 - y_{ijk}) - H_{ij} * y_{ijk} \geq 0 \ \ \forall i, j \in N, \forall k \in K, \forall l \in \{L/1\} \tag{5}$$

$$S_{ikl} \geq (k-1) * T \ \ \forall k \in K \tag{6}$$

$$\sum_{I} \sum_{L} P_i * x_{ikl} + \sum_{I} \sum_{J} H_{ij} * y_{ijk} \leq T * z_k \ \ \forall k \in K \tag{7}$$

$$C_{max} \geq C_{ikl} \ \ \forall i \in N, \forall k \in K, \forall l \in L \tag{8}$$

$$x_{ikl}, y_{ijk}, z_k \in \{0, 1\}, S_{ikl} \geq 0, C_{ikl} \geq 0 \tag{9}$$

In the objective function, the goal is to minimize total time that is required to finish all jobs, in other words to minimize makespan. The first constraint (1) is a constraint that enables each jobs to have just a shift and a sequence in assignment. The second constraint (2) ensures that every sequence in all shifts have at most one job. In the third constraint (3), if some jobs are assigned to successive sequence in any shift, binary decision variable (y_{ijk}) must be 1. The constraint (4) shows the relationship between completion time and starting time of any job. The constraint (5) calculates starting time of any job depends on the completion time of previous job. In this constraint, sequence dependent setup times are taken into consideration to calculate starting time of next job in a shift if there is a job before that. The constraint (6) enables the starting time of any job in first sequence in any shift to be determined as multiples of shift times. The constraint (7) is capacity constraint that enables assigned jobs to any shift to not exceed the total time capacity of that shift. The constraint (8) ensures that makespan equals to the greatest completion time among all jobs. The last constraint (9) is about binary and sign restrictions.

This model is run in Spyder which is scientific python development environment, by using Gurobi software. The results show that the model gives an optimal solution for the instances with up to 9 jobs. Real-life instances which typically have

100 jobs cannot be solved using this mathematical model. In high volume business environments, some heuristic algorithms can be used to overcome the problem.

CHAPTER IV

PROPOSED SOLUTION APPROACH

In this chapter, proposed solution approaches are presented. The problem is thought as bin packing problem. Firstly, FFD algorithm in the literature is implemented to the investigated problem. The FFD algorithm implementation process is explained in all stages. After the implementation of FFD algorithm, the optimality gap is examined to see if more improvement can be achieved with new approaches. It turns out that new heuristic approaches need to be proposed due to high optimality gap. Finally, new heuristic approaches are proposed, their working principles are explained and the pseudocodes of proposed algorithms are demonstrated. Moreover, small scaled example of IFFH is presented.

It has been mentioned that FFD algorithm provides better solution by comparison with FF random sequencing for the bin packing problems in the literature [6]. We tackled our problem as a bin packing problem, and we applied FFD algorithm to the problem. It was seen that the improvement of solution which is provided by FFD algorithm in comparison with FF algorithm is not enough, because there is still a huge optimality gap.

The optimality gap is percentage difference between completion time of all jobs and predetermined lower bound. The main objective of new heuristic approaches is to minimize optimality gap as much as possible. There are two main reasons for huge optimality gap, these are succession of jobs which have high setup times between each other, and idle time of machine. New heuristic approaches focuses on to decrease them as much as possible.

The problem is thought as bin packing problem, because there are requirements

for shifts like bins, it is wanted to keep at minimum number and full capacity as much as possible. IFFH algorithm starts with result of FFD Algorithm, because FFD algorithm gives smallest worst case ratio according to other bin packing algorithms [6]. Also, it is demonstrated that FFD algorithm enables shortest time to complete all jobs in comparison with FF when there is not any setup times among jobs [10].

FFD algorithm begins with getting processing times and sequence dependent setup times from user. Processing times are sorted with descending order. Assignment of jobs begins with first job in sorted list. It is thought that there are possible number of shifts as much as number of jobs. For each shift, assignment process is carried out by starting with first job in sorted list. The assignment process is done as follows. If the processing time of present job is less than the idle time of shift, the job is assigned to that shift. The idle time is calculated as subtracting total processing times of all jobs in shift, total sequence dependent setup times between these jobs and setup time between last job in shift and present job, from total length of shift. After the assignment processes, completion time of last job is kept in memory. The result of FFD includes randomized sequence dependent setup times, so it gives an opportunity to decrease them. Firstly, shifts are investigated individually to provide minimum total setup times. In order to get minimum total setup times in a shift, it is obligation to calculate all permutations of jobs, but it is impossible to calculate when the number of jobs is high. Therefore, a heuristic approach is required to close to optimum solution. We developed heuristic algorithms to overcome above problems.

4.1 Improved First Fit Heuristic Algorithm

In this section, implementation processes of IFFH is explained in all stages. Pseudocodes of new heuristic algorithm is demonstrated with main steps and small scaled example of IFFH algorithm is presented in different subsections.

It begins with getting the results of FFD algorithm. After the getting results

of FFD algorithm, the following operations are done for each shift. For all possible binary permutations of jobs in each shift, setup times are ranked, then two jobs which have minimum setup times between each other are chosen to place to first two sequence in shift. If there are more than two jobs in shift, the job with the minimum setup time between second jobs in shift is placed to third sequence, and goes on placing till to finish all jobs in shift. The replacement between jobs enables to decrease used setup times in a shift, so it creates idle time for shift. In order to fill idle time of shifts, all shifts are concatenated with this order and FF algorithm is applied. This process is repeated until there is no more improvement for minimizing completion time of all jobs. Main steps of Pseudocodes of IFFH algorithm is as following:

4.1.1 Pseudocodes of Improved First Fit Heuristic Algorithm

1. Input shifts //Get the results of FFD algorithm

2. For each shift, find the minimum setup time between jobs in binary permutations. Then, place two jobs which has minimum dependent setup time to first two sequence.

3. If there are more than two jobs in the shifts, find the job which has minimum dependent setup time after the second job, then place it third place. After that, continue to place all jobs with same logic.

4. Concatenate all shifts.

5. Start to assign jobs to shifts with this order

6. For each shift in the range of n:

    For each job in the range of n:

        If the job is not assigned to any shift:

            If Processing time of present job $\leq$Length of present shift-(Total processing times of previous jobs in the shift+ Total dependent setup times between previous jobs in the shift+ dependent setup time between present job and last job in

the shift), assign the present job to present shift.

7. Calculate the completion time of last job. Keep it in memory. Return to 2.

8. Repeat the algorithm (2-6) until to get no more improvement, and accept the minimum completion time of all jobs as a solution.

9. Compare the solution with FFD Algorithm

4.1.2 Small Scaled Example of Improved First Fit Heuristic Algorithm

In this section, small scaled example of Improved First Fit Heuristic Algorithm is presented with figures. Firstly, FFD algorithm is run with small number of jobs, and completion times are calculated. Then, IFFH algorithm is run, and results are shown with figures.

An example was created by taking into consideration 10 jobs. Length of shift was taken 20 minutes in this example. Processing times of jobs, sequence of jobs and sequence dependent setup times between jobs is shown below.

Table 1: Processing Times and Sequences of Jobs

Pi	7	5	2	9	5	10	8	9	1	9
i	1	2	3	4	5	6	7	8	9	10

Table 2: Sequence Dependent Setup Times

H_{ij}	1	2	3	4	5	6	7	8	9	10
1	M	1	1	3	3	1	1	3	1	2
2	1	M	1	2	3	2	3	1	3	1
3	3	3	M	1	1	3	3	3	3	2
4	2	2	2	M	3	2	2	1	1	1
5	3	1	2	2	M	2	3	2	2	1
6	3	1	1	1	3	M	3	3	3	3
7	2	3	2	2	2	2	M	1	2	1
8	3	2	3	2	3	3	2	M	3	1
9	1	1	1	2	2	3	1	3	M	1
10	1	2	1	3	2	2	3	3	1	M

In order to implement FFD algorithm, the jobs were ranked in descending order,and assignment was carried out. In the first figure, results of FFD algorithm on example is shown. Assignment of jobs to shifts and used setup times are demonstrated.

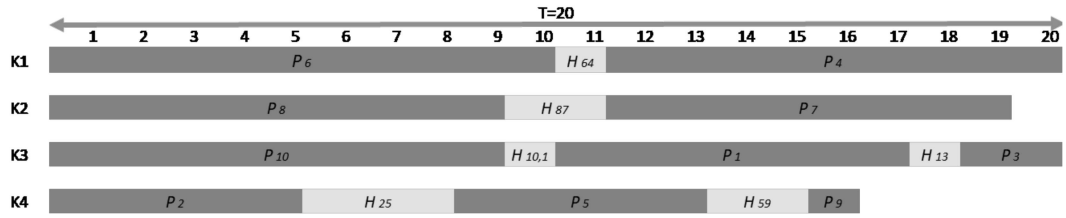

Figure 1: Example Solution of FFD Algorithm

When FFD algorithm was applied, the completion time of all jobs has been 76 minutes, and 4 shifts were used for all jobs.

In the second figure, according to results of FFD algorithm, sequence replacements were done for each shift. This is the beginning of the IFFH algorithm. As it seen in the figure, there was shortening in second and fourth shift in terms of used times.

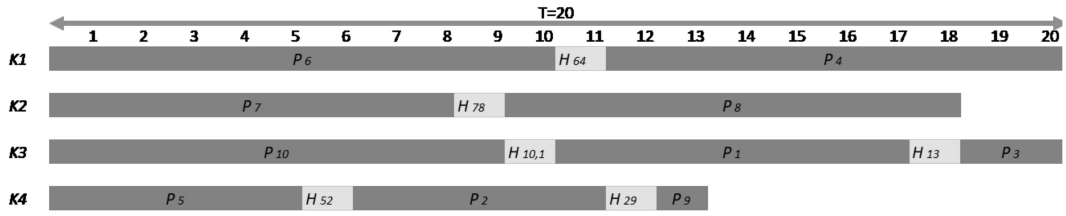

Figure 2: New Solution obtained by IFFH Algorithm

In order to fill in the gaps in shifts, FF algorithm is applied for new sequence of jobs which is result of replacements in shifts. In the third figure, the final solution of IFFH algorithm is shown. After the replacements of jobs in the shifts, FF algorithm was applied for new sequence which is created by concatenating shifts.

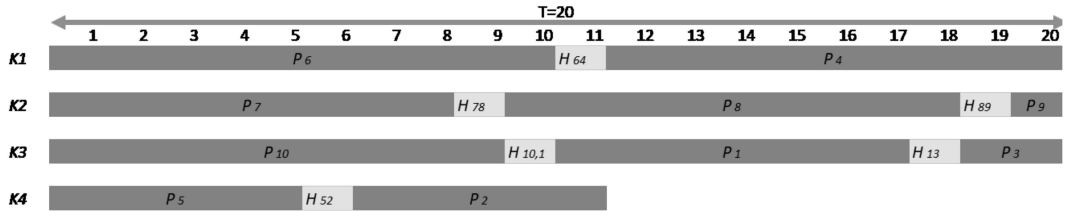

Figure 3: Final Solution

When IFFH algorithm was applied, it was seen that 9_{th} job in the last shift passed to gap in the second shift and the completion time of all jobs has been 71 minutes. In the investigated cases, the algorithm continues to run iteratively, but for this example,there is not better solution than final solution.

4.2 Improved First Fit Heuristic-2 Algorithm

In this section, implementation processes of Improved First Fit Heuristic-2 algorithm is explained in all stages. Then, pseudocodes of the algorithm is demonstrated with main steps in subsection.

We further modified the IFFH as follows: Shifts are partially handled in order to enable more improvement on heuristic approach. Binary combinations of shifts

without last shift and last shift in the result of IFFH algorithm are concatenated. FFD and IFFH algorithms are applied three times on for each combination of triple shifts, and the minimum completion times are kept in memory. Main steps of Pseudocodes of IFFH-2 algorithm is as following:

4.2.1 Pseudocodes of Improved First Fit Heuristic-2 Algorithm

1. Get the results of IFFH algorithm.
2. For the best solution in IFFH algorithm, find binary combinations of shifts without last shift
3. Concatenate them with last shift individually.
4. For each concatenated shift groups, apply FFD and IFFH algorithm 3 times.
5. Calculate completion time for each concatenated shift groups, keep them in memory.
6. Take the minimum completion time in concatenated shift groups, then add it to total completion time of other shifts. Accept it as a solution.
7. Compare the solution with IFFH algorithm.

Pseudocodes of algorithms were explained above, which are informal way of description of what we proposed as a heuristic algorithm. Pseudocode of algorithms have been converted into a mathematical models in Python. These mathematical model was run on Gurobi, results were compared with each other.

CHAPTER V

COMPUTATIONAL RESULTS

In this chapter, computational results of proposed approaches are shown. Firstly, existing model is run, it is seen that the software does not give solution with high number of jobs. Some heuristic algorithms in the literature are explained, and FFD algorithm is taken as open-to-development algorithm. Then, instances are created with high number of jobs to implement algorithms on. After implementation of algorithms on different data sets, the comparison table are created to see if more improvements are achieved obviously. Then, inferences about comparison table of computational results are made. Finally, runtimes of algorithms are compared with each other, and inferences about that is made.

The investigated problem is in the class of mixed integer programming problems. There are several software to solve these kind of problems. In this study, Gurobi software which is based on Python language was chosen to solve the problem.

Firstly, existing model was run on software, which demonstrated that if there are high number of jobs, existing model does not give a solution. Because the software needs a lot of time to solve the problem for high numbers of jobs. When we used data set with more than 8 jobs, the software could not give a solution in reasonable time. The reason is that there are 9 factorial feasible solution (which is 362.880) and the software have to look all of them. Therefore, we had to terminate the model.

In the literature, there are some heuristic algorithms such as NF, BF and FF for bin packing problems. Many different algorithms have been created based on these basic algorithms such as BFD and FFD. NF algorithm works on this principle: Put each item into the last bin which it fits. BF algorithm works on this principle: Put

each item into the oldest bin which has minimum slack. FF algorithm works on this principle: Put each item into the oldest bin which it fits. BFD algorithm is a heuristic algorithm that the items are sorted in descending order. It works with BF algorithm principle. FFD algorithm is a heuristic algorithm that the items are sorted in descending order. It works with FF algorithm principle. FFD algorithm enables last shift to have minimum weight. The objective of existing problem is already minimizing total completion time of all jobs. Therefore, FFD algorithm is taken as open-to-development algorithm for existing problem in the literature.

All above heuristic approaches works without setup times, but in the investigated problem there are dependent setup times between jobs. Assignment of a new job to any shift depends on the last job in that shift. Setup times between jobs can be considered as a cost. It is significant how that the cost can be decreased to improve solution. In this thesis, a new heuristic method is suggested to improve solution.

In the real life, there are huge single machine scheduling problems. We are interested in that kind of problem which has high number of jobs. We know that the problems is NP-hard and they cannot be solved in polynomial time. We developed a heuristic algorithm, and we applied it on different data sets. The data sets from 50 jobs to 500 jobs were generated (number of jobs= 50,100,150,200,250,300,350,400,450,500). The processing times of jobs were randomly distributed between 1 min. and 60 min. [1, 60]. The length of a shift is taken as 480 minutes. For each data set for number of jobs, three different setup time distribution were randomly generated, which are [1, 5], [1, 10] and [6, 10]. For each set, 15 instances were randomly generated. Therefore, there were 450 instances to see performance of existing and developed heuristic algorithms.

In order to measure performance of heuristic algorithms, we looked at gaps between lower bounds and make-span. FFD, IFFH and IFFH-2 algorithms were compared in terms of gaps, then table 1 is appeared.We used the processor of Intel(R)

Core(TM) i5-6200U CPU @ 2.30GHz, 2400 Mhz with 8,00 GB RAM to measure performances of heuristic algorithms.

The first column of table shows the number of jobs which are used for instances. The second column of table shows distribution range of setup times. Other columns of table shows the optimality gaps of FFD, IFFH and IFFH-2 algorithms respectively. First average in the table shows the average gaps of instances which have setup times in the range of [1, 5]. Second average in the table shows the average gaps of instances which have setup times in the range of [1, 10]. Third average in the table shows the average gaps of instances which have setup times in the range of [6, 10]. Overall average in the table shows the average gaps of all instances.

The lower bound (LB) for optimality gap is calculated with sum of total processing time (TPT) and minimum total possible setup times (MTPS). Minimum total possible setup time is calculated as in below.

$$MTPS = (n - [(\sum_{I} P_i/T)] * (min(H_{ij}))\ \ \forall i, j \in N \tag{10}$$

The optimality gap is percentage difference between completion time of all jobs that means upper bound (UB), and predetermined lower bound for each algorithm.

$$OptimalityGap = (UB - LB)/LB \tag{11}$$

Table 3: Comparison of Gaps

n	H_{ij}	FFD	$IFFH$	$IFFH-2$
50		5.76%	2.02%	1.73%
100		5.95%	1.69%	1.46%
150		5.86%	1.62%	1.4%
200		5.91%	1.55%	1.33%
250		5.61%	1.52%	1.34%
300	[1,5]	5.63%	1.5%	1.35%
350		5.66%	1.45%	1.28%
400		5.65%	1.44%	1.3%
450		5.66%	1.47%	1.35%
500		5.64%	1.49%	1.38%
Average		5.73%	1.58%	1.39%
50		13.15%	4.81%	4.15%
100		13.14%	4.1%	3.79%
150		12.3%	3.94%	3.43%
200		12.66%	3.99%	3.56%
250		12.69%	3.86%	3.53%
300	[1,10]	12.52%	3.83%	3.49%
350		12.33%	3.82%	3.51%
400		12.54%	3.84%	3.59%
450		12.57%	3.85%	3.6%
500		12.62%	3.77%	3.57%
Average		12.65%	3.98%	3.62%
50		4.9%	1.95%	1.55%
100		4.93%	1.81%	1.44%
150		4.83%	1.54%	1.3%
200		4.8%	1.55%	1.33%
250		4.85%	1.55%	1.37%
300	[6,10]	4.75%	1.46%	1.31%
350		4.74%	1.47%	1.34%
400		4.84%	1.43%	1.3%
450		4.69%	1.45%	1.34%
500		4.72%	1.51%	1.41%
Average		4.81%	1.57%	1.37%
Overall Average		7.73%	2.37%	2.13%

As it is shown table, it is obvious that proposed heuristic algorithms enables all jobs to finish in shorter time than FFD algorithm for all instances. Because, both IFFH and IFFH-2 algorithms have low optimality gap in comparison with FFD algorithm.

It is demonstrated that IFFH algorithm gives better solution than FFD algorithm in overall because of having lower gap. The IFFH algorithm provides a minimum of 2.95%, a maximum of 9.04% and an average of 5.36% better solutions than the FFD algorithm. The IFFH-2 algorithm provides a minimum of 3.32%, a maximum of 9.35% and an average of 5.6% better solutions than the FFD algorithm. The IFFH-2 algorithm provides a minimum of 0.1%, a maximum of 0.66% and an average of 0.25% better solutions than the IFFH algorithm.

When the setup time distribution is taken between 1 minute and 5 minutes, gaps between lower bound and makespan are shown at the top of the table. It is seen that IFFH and IFFH-2 gaps are 1.58% and 1.39% on average respectively. They are 4.16% and 4.34% better than FFD algorithm respectively.

When the setup time distribution is taken between 1 minute and 10 minutes, gaps between lower bound and makespan are shown in the middle of the table. It is seen that IFFH and IFFH-2 gaps are 3.98% and 3.62% on average respectively. They are 8.67% and 9.03% better than FFD algorithm respectively.

When the setup time distribution is taken between 6 minutes and 10 minutes, gaps between lower bound and makespan are shown at the bottom of the table. It is seen that IFFH and IFFH-2 gaps are 1.57% and 1.37% on average respectively. They are 3.24% and 3.44% better than FFD algorithm respectively.

Considering all instances, it is demonstrated that when the variability of setup times increase, proposed heuristic algorithms give better solution. While setup times are between 1 minute and 10 minutes, heuristic algorithms give 8.67% and 9.03% better solution on average respectively, which is much better than while setup times

are between 1 minute and 5 minutes, or 5 minutes and 10 minutes. Besides, when the variability of setup times is same, heuristic algorithms gives better solution for instances which have low setup times.

As it explained before we worked with 450 instances, and for all instances IFFH and IFFH-2 algorithms give better solutions than FFD algorithm. However, in comparison of IFFH and IFFH-2 algorithms, for 448 instances IFFH-2 algorithm gives better solution than IFFH algorithm. Two instances which have not better solutions were encountered on data sets which have 50 number of jobs.

Table 4: Runtime in seconds.

n	FFD	$IFFH$	$IFFH-2$
50	0.09	1.06	1.34
100	0.12	3.57	5.07
150	0.28	7.36	12.91
200	0.54	13.45	20.84
250	0.83	21.14	34.34
300	1.18	30.01	47.32
350	1.59	39.43	71.93
400	1.97	50.45	94.36
450	2.68	69.94	126.57
500	3.17	87.22	139.79

In the above table, run times of FFD algorithm and proposed algorithms are demonstrated in seconds. While number of jobs increase, run times of algorithms are getting high. FFD algorithm runs 1.24 seconds on average. IFFH algorithm runs 32.36 seconds on average and IFFH-2 algorithm runs 55.44 seconds on average. IFFH-2 algorithm needs on average 71% more time than IFFH algorithm.

CHAPTER VI

CONCLUSION

In this chapter, the main conclusions are explained about new heuristic algorithms, and suggestions are made for future works.

In this thesis, we studied on single machine scheduling with dependent setup times. Firstly, we analyzed the problem and existing model was generated. We explained that the problem is NP-hard and it cannot be solved in polynomial time. Literature researches was done, and solution methods of similar problems are investigated. We are inspired by Gonzalez and Framinan's article that they addressed the single machine scheduling problem with periodic maintenance as bin packing problem.

We solved the problem with First Fit Decreasing algorithm, calculated gaps between lower bound and make-span. We decided to develop new heuristic algorithms which provides us to have lower gaps. According to results of FFD algorithm, there was idle times in shift and much used setup times. Our objective was to decrease useless times, so we investigated shifts individually and provided shorter setup times with replacement of jobs in themselves. Our replacement strategy was placing two jobs that have minimum setup times between each other to first two sequence, and after second job in shift, placing any job which has minimum dependent setup times, then goes on placing other jobs with same method. Then, we concatenated shifts and run FFD algorithms again till to getting not improvement. Improved first fit heuristic algorithm provided us 5.36% better solution on average in comparison with FFD algorithm which has 7.73% gap on average.

In order to decrease gap much more, we developed Improved First Fit Heuristic-2 algorithm. It works on the same principle with IFFH algorithm, the difference was

that it runs on triple shifts. Triple shifts were composed of last shift and any other two shifts. The triple shifts which provide maximum decreasing for make-span was accepted and by adding completion time of this triple shifts to completion time of other shifts, make-span is calculated. The IFFH-2 algorithm provided us 0.25% better solution on average in comparison with IFFH algorithm.

It was obviously demonstrated that the new heuristic algorithms gives better solution in comparison with FFD algorithm in the literature. Moreover, the results demonstrated that when the variability of setup times increase, heuristic algorithms give better solutions, and at the same variability heuristic algorithms give better solution for shorter setup times.

For future work, we can improve some algorithms to determine lower bounds more precisely. Completion times for each job can be taken into consideration individually and some delay costs can be considered. The processing times are deterministic but for some real issues stochastic modelling can be done.

Bibliography

[1] A. Allahverdi, N.D. Gupta and T. Alwodosian "Review of scheduling research involving setup considerations", *Omega, International Journal of Management Science*, vol. 27, pp. 219-239, 1999.

[2] J. E. C. Arroyo, G. V. P. Nunes and E. H. Kamke, "Iterative local search heuristic for the single machine scheduling problem with sequence dependent setup times and due dates", *Ninth International Conference on Hybrid Intelligent Systems*, vol. 1, pp. 505–510, 2009.

[3] K. R. Baker and M. J. Magazine, "Minimizing maximum lateness with job families", *European Journal of Operational Research*, vol. 127(1) pp. 126–39, 2000.

[4] L. P. Bigras, M. Gamache, and G. Savard, "The Time-Dependent Travelling Salesman Problem and Single Machine Scheduling Problems with Sequence Dependent Setup Times", *Discrete Optimization*, vol. 5, pp. 685-699, 2008.

[5] D. Biskup and T. C. Edwin Cheng, "Single-Machine Scheduling With Controllable Processing Times and Earliness, Tardiness and Completion Time Penalties", *Engineering Optimization*, vol. 31:3, pp 329-336, 1999.

[6] E. G. Coffman, M. R. Garey, and D. S. Johnson, "Approximation Algorithms for Bin-Packing - An Updated Survey" *CISM International Centre for Mechanical Sciences*, pp. 49-106, 1984.

[7] C. Gagne, W. L. Price and M. Gravel, "Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times", *Journal of the Operational Research Society*, vol. 53, pp. 895–906, 2002.

[8] J. W. Gavett, "Three Heuristic Rules for Sequencing Jobs to a Single Production Facility", *Management Science*, vol. 11(8), pp. B-166-B-176, 1965.

[9] C. R. Glassey, "Minimum Change-Over Scheduling of Several Products on One Machine", *Operations Research*, vol. 16(2) pp. 342-352, 1968.

[10] P. P. Gonzalez and J. M. Framinan, "Single machine scheduling with periodic machine availability", *Computers Industrial Engineering*, vol. 123, pp. 180-188, 2018.

[11] L. Guimarães, D. Klabjan, and B. Almada-Lobo, "Annual production budget in the beverage industry", *Engineering Applications of Artificial Intelligence*, vol. 25(2), pp. 229-241, 2012.

[12] S. K. Gupta and J. Kyparisis, "Single Machine Scheduling Research", *Omega, International Journal of Management Science*, vol. 15(3) pp. 207-222, 1987.

[13] J. Leiva and V. Albornoz, "Short-term Production Scheduling in the Soft Drink Industry", *In Proceedings of 5th the International Conference on Operations Research and Enterprise Systems*, pp. 416-423, 2016.

[14] D. P. Ronconi and M. S. Kawamura, "The single machine earliness and tardiness scheduling problem: lower bounds and a branch-and-bound algorithm", *Computational Applied Mathematics*, vol. 29(2), pp. 107–124, 2010.

[15] J. Shwimer, "On the N-Job One-Machine, Sequence-Independent Scheduling Problem with Tardiness Penalties: A Branch-Bound Solution", *Management Science*, vol. 18(6) pp. B-301-B-313, 1972.

[16] F. Sourd and S. K. Sidhoum, "The One-Machine Problem with Earliness and Tardiness Penalties", *Journal of Scheduling*, vol. 6, pp. 533-549, 2003.

[17] X. Sun, J. S. Noble and, C. M. Klein, "Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness", *IIE Transactions*, vol. 31, pp. 113-124, 1999.